Зборник радова Факултета техничких наука, Нови Сад

УДК: 007:004.056
ДОИ: https://doi.org/10.24867/35BE14Stanisic

Аутентификација и ауторизација базиране на сесијама и токенима за постојеће информационе системе

Session and Token Based Authentication and Authorization for Legacy Information Systems

Живко Станишић, *Факултет техничких наука, Нови Сад*

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *Рад представља интеграцију OAuth 2.0 и OpenID Connect у постојеће апликације са сесијама, уз AWS Cognito систем за управљање корисницима и токенима. Предложени хибридни модел са .NET доказом концепта описује архитектуру, верификацију/опозив токена и креирање сесије, омогућавајући постепену миграцију ка cloud стандардима без реконструкције.*
Кључне речи: *OAuth 2.0, OpenID Connect, AWS Cognito, сесије, .NET*

Abstract – *This thesis presents the integration of OAuth 2.0 and OpenID Connect into legacy session-based applications, using AWS Cognito for user and token management. The proposed hybrid model, supported by a .NET proof of concept, describes the system architecture, token verification and revocation, and session creation, enabling a gradual migration to cloud standards without full system reconstruction.*

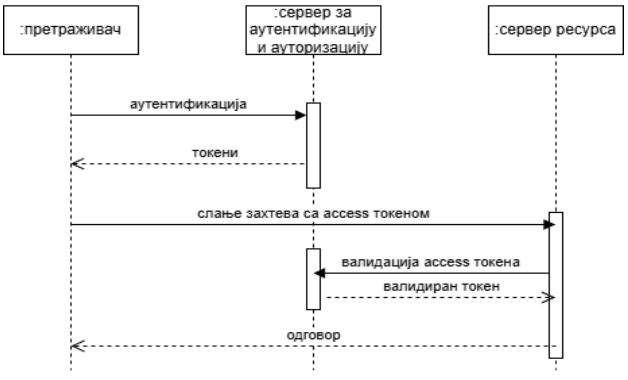
Keywords: *Auth 2.0, OpenID Connect, AWS Cognito, sessions, .NET*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Горан Сладић, ред. проф.

1. УВОД

Уз ширење пословања на интернету јављају се изазови за старе системе, па предузећа често примењују хибрид *legacy* система са микросервисима у облаку [1]. У овом решењу аутентификација/ауторизација се ослања на *AWS Cognito* и *OAuth* токене [2]. Недостајући подаци се држе у серверској сесији повезаној са *access* токеном, а *refresh* токени у *DynamoDB* се могу опозвати при промени креденцијала/суспензији налога. Тако се испоштује *OAuth* и уз минималне измене постојећег кода добија практично, економично и лако прошириво решење за остатак система. У овом раду се приказује једно хибридно решење за аутентификацију и ауторизацију које омогућава интеграцију савремених *cloud* механизама са постојећим апликацијама које користе сесије. Циљ рада је да се прикаже како се употребом *OAuth 2.0* и *OpenID Connect* стандарда може проширити постојећи сесијски систем без потпуне реконструкције. На тај начин постиже се модернизација постојећих апликација и њихово усклађивање са савременим безбедносним стандардима. На слици 1 приказан је дијаграм секвенци протока у протоколу *OAuth 2.0*, који илуструје комуникацију између три главне компоненте система: претраживача, сервера за аутентификацију и ауторизацију и сервера ресурса.

1. **Аутентификација корисника** - процес почиње када корисник преко претраживача шаље захтев серверу за аутентификацију и ауторизацију, где се врши провера његовог идентитета.
2. **Издавање токена** - након успешне аутентификације, сервер враћа клијенту један или више токена (најчешће *access* и *refresh* токени) [4].
3. **Слање захтева ка серверу ресурса** - претраживач затим шаље *HTTP* захтев серверу ресурса, при чему у заглављу захтева шаље и добијени *access* токен.
4. **Валидација токена** – сервер ресурса прослеђује *access* токен серверу за аутентификацију и ауторизацију ради провере његове исправности и важења.
5. **Одговор ка клијенту** - ако је токен валидан, сервер ресурса враћа тражени одговор, а у супротном приступ је одбијен.



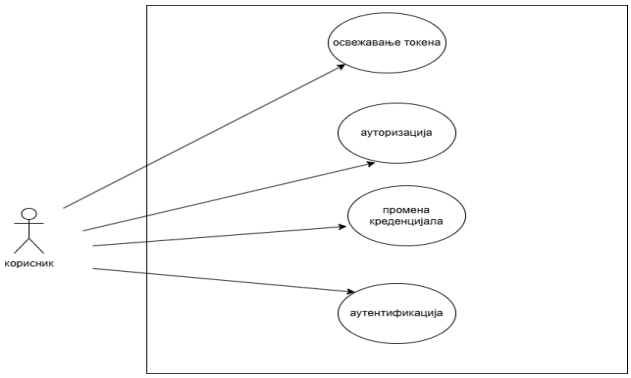
Слика 1. Дијаграм секвенци OAuth 2.0

2. ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА

Комбиновање сесија и токена задржава проверену интеракцију у претраживачу и обезбеђује интероперабилност са спољним провајдерима идентитета и *API*-јима у оквиру исте архитектуре. Обрасци се углавном разликују по месту чувања токена и где се терминише сесија, а основни циљ остаје исти [5]. Неки од образаца су: *Cookie Sessions and OpenID Connect*, *Legacy Monolith with Sessions and Federated Login*, *Backend-for-Frontend*, *API Gateway Token-to-Session Bridge* и *SaaS: App Session and Per-Provider OAuth Connections* [6][7][8].

3. МОДЕЛ СИСТЕМА

Поглавље укратко представља најчешће случајеве моделоване *UML* дијаграмима, са актерима: ауторизациони сервер, сервер ресурса и актера који их повезује. На слици 2 приказан је дијаграм случајева коришћења најчешћих сценарија коришћења система за аутентификацију и ауторизацију. Главни актер је корисник, који може да изврши више основних операција у оквиру система као што су: аутентификација, ауторизација, освежавање токена и промена креденцијала.

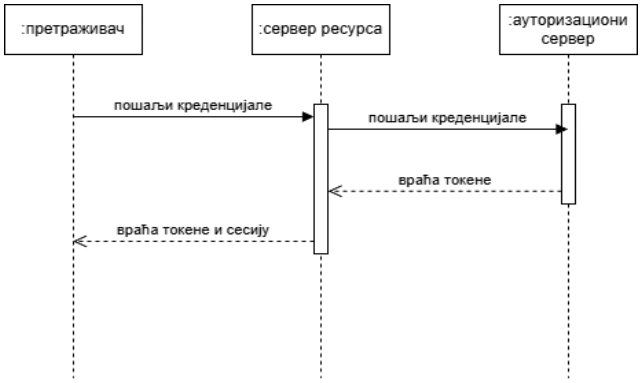


Слика 2. Приказ најчешћих случајева коришћења

3.1. Аутентификација

Аутентификација у овом решењу подразумева проверу валидности креденцијала и издавање *ID, access* и *refresh* токена у *JWT* формату [9]. Корисник шаље *email* и лозинку серверу ресурса, који преко ауторизационог сервера валидира креденцијале, добија токене, креира сесију и све заједно враћа претраживачу. На слици 3 приказан је дијаграм секвенци процеса аутентификације који илуструје размену података између претраживача, сервера ресурса и ауторизационог сервера.

1. **Слање креденцијала** - корисник преко претраживача уноси своје креденцијале који се шаљу ка серверу ресурса.
2. **Прослеђивање ка ауторизационом серверу** – сервер ресурса прослеђује креденцијале ауторизационом серверу који је задужен за проверу идентитета.
3. **Издавање токена** - након успешне провере ауторизациони сервер враћа токене серверу ресурса.
4. **Враћање токена и сесије клијенту** – сервер ресурса прослеђује добијене токене и информације о сесији назад претраживачу, чиме корисник постаје аутентификован и може приступати заштићеним ресурсима.

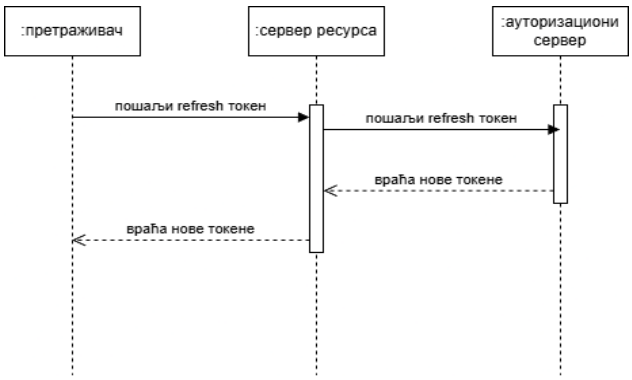


Слика 3. Дијаграм секвенци аутентификације

3.2. Освежавање токена

ID и *access* токени су кратког века, док *refresh* токен важи знатно дуже у складу са потребама пословања. Пре истека краткотрајних токена, клијент преко сервера ресурса шаље *refresh* токен ауторизационом серверу, који по валидацији издаје нове *ID* и *access* токене. На слици 4 приказан је дијаграм секвенци процеса освежавања токена, који описује како клијент добија нове токене.

1. **Слање *refresh* токена** - претраживач шаље *refresh* токен серверу ресурса како би добио нове токене без поновне пријаве.
2. **Прослеђивање ка ауторизационом серверу** – сервер ресурса прослеђује *refresh* токен ка ауторизационом серверу ради валидације.
3. **Издавање нових токена** - након успешне провере *refresh* токена, ауторизациони сервер враћа нове токене серверу ресурса.
4. **Враћање нових токена клијенту** - сервер ресурса прослеђује нове токене претраживачу, чиме се корисникова сесија продужава без потребе за поновним уносом креденцијала.



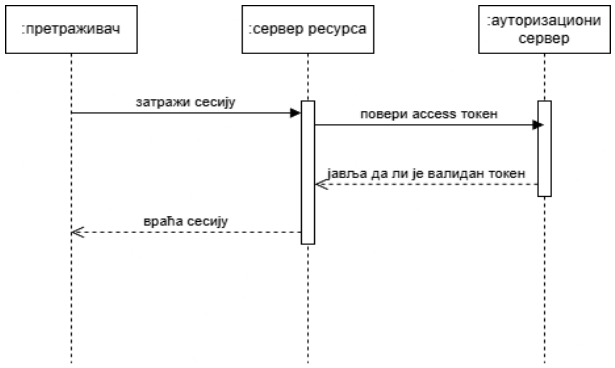
Слика 4. Дијаграм секвенци освежавање токена

3.3. Ауторизација

Ауторизација у овом решењу је сложенија јер се у постојећи сесијски систем уводи ауторизациони сервер. Сесија се логички дели тако да *access* токен

преузима улогу кључа за ауторизацију, а *ID* токен замењује сесијски објекат на клијенту. Због еволуције наслеђене апликације серверска сесија је дубоко укорењена у код, па клијент више нема потребу за сесијом док сервер и даље зависи од ње, што намеће хибридно повезивање сервера ресурса и ауторизационог сервера. На слици 5 приказан је дијаграм секвенци процеса ауторизације, који илуструје ток комуникације између претраживача, сервера ресурса и ауторизационог сервера током провере приступа.

1. **Захтев за сесију** - корисник преко претраживача шаље захтев серверу ресурса ради приступа заштићеним подацима.
2. **Провера *access* токена** – сервер ресурса шаље добијени *access* токен ка ауторизационом серверу ради провере његове валидности.
3. **Одговор о валидности токена** - ауторизациони сервер враћа резултат провере, тј. да ли је токен валидан и да ли корисник има потребна права приступа.
4. **Враћање сесије клијенту** - уколико је токен исправан, сервер ресурса враћа кориснику активну сесију и дозвољава приступ ресурсима.

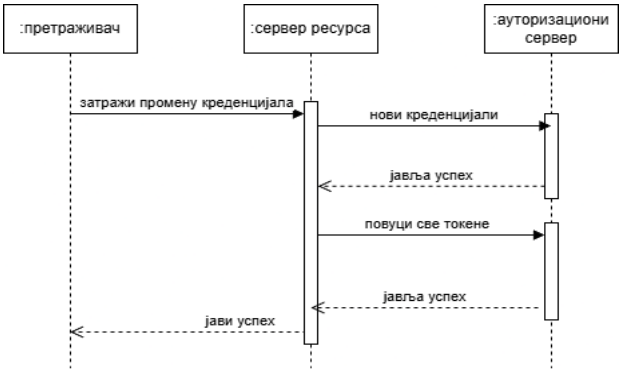


Слика 5. Дијаграм секвенци ауторизације

3.4. Промена креденцијала

Промена креденцијала може довести до озбиљних пропуста ако се не обраде сви сценарији, нарочито кад корисник није улогован. Нападач може остати пријављен, злоупотребити још важећи *access* токен за креирање сесије или освежити приступ помоћу *refresh* токена. Решење је инвалидирање свих токена након промене креденцијала где сервер ресурса шаље идентификатор корисника ауторизационом серверу, који повлачи токене. Исти поступак важи и за деактивацију корисника. На слици 6 приказан је дијаграм секвенци промене креденцијала који приказује кораке комуникације између претраживача, сервера ресурса и ауторизационог сервера током ажурирања података за пријаву.

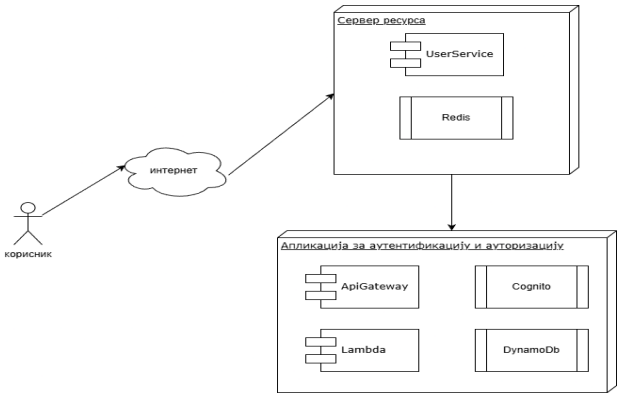
1. **Захтев за промену креденцијала** - корисник преко претраживача шаље захтев серверу ресурса за промену креденцијала.
2. **Прослеђивање нових креденцијала** – сервер ресурса шаље нове креденцијале ауторизационом серверу ради ажурирања података.
3. **Потврда успеха** - ауторизациони сервер враћа поруку о успешној промени креденцијала.
4. **Повлачење постојећих токена** - након успешног ажурирања, сервер ресурса шаље захтев за поништавање свих постојећих токена који су били повезани са старим креденцијалима.
5. **Коначна потврда успеха** - ауторизациони сервер потврђује повлачење токена, након чега сервер ресурса обавештава клијента да је процес успешно завршен.



Слика 6. Дијаграм секвенци промене креденцијала

4. ИМПЛЕМЕНТАЦИЈА

Имплементација се састоји из два дела: *TypeScript/Node.js* апликације за аутентификацију и ауторизацију на *AWS*-у (*Cognito, DynamoDB, API Gateway, Lambda*) која примењује микросервисну архитектуру и решава скалабилност и једноставне ресурсне *.NET* апликације са сесијама у *Redis*-у. На слици 7 приказан је *deployment* дијаграм целог система. Корисник преко интернета приступа серверу ресурса који чине сервиси, као што је *UserService*, који је део *.NET Web API*-ја и *Redis* база података намењена за чување сесија. Сервер ресурса има приступ апликацији за аутентификацију и ауторизацију, која користи поменуте *Cognito, DynamoDB, API Gateway* и *Lambda* сервисе [10][11].



Слика 7. Приказ *deployment* дијаграма

У наставку је дат једноставан пример који приказује два повезана елемента процеса креирања сесије са ауторизацијом. На слици 8 приказана је

имплементација функције *action* која врши проверу ваљаности *access* токена у процесу ауторизације.

```
export async function action(
    request: ValidateAccessTokenRequest
): Promise<ValidateAccessTokenResponse> {
    const provider = new CognitoIdentityProvider({
        region: validateAndGetRegion(),
    });
    const user = await provider.getUser({
        AccessToken: request.accessToken,
    });

    if (!!user.Username && user.Username !== '') {
        return { isValid: true };
    }

    return { isValid: false };
}
```

Слика 8. Приказ акције ауторизације

На слици 9 приказана је метода *CreateSession*, која омогућава креирање нове сесије за већ аутентификованог корисника. Ова метода позива горе поменуту акцију да провери да ли је *access* токен валидан.

```
public async Task<UserSession> CreateSession()
{
    var user = _httpContextAccessor.HttpContext?.User;
    if (user == null || !user.Identity?.IsAuthenticated == true)
    {
        throw new UnauthorizedException();
    }

    var email = user.FindFirst("username")?.Value;
    var systemUser = _repo.GetByEmail(email)
        ?? throw new KeyNotFoundException("User not found.");
    var authHeader = _httpContextAccessor.HttpContext?.Request
        .Headers[HeaderNames.Authorization].FirstOrDefault();

    var token = authHeader != null &&
        authHeader.StartsWith("Bearer ", StringComparison.OrdinalIgnoreCase)
        ? authHeader["Bearer ".Length..].Trim()
        : authHeader;

    var accessTokenResponse = await _authServiceClient
        .ValidateAccessToken(new ValidateAccessTokenRequest
    {
        AccessToken = token
    });

    if (accessTokenResponse == null || !accessTokenResponse.IsValid)
    {
        throw new UnauthorizedException();
    }

    return await _sessionService.CreateAsync(systemUser);
}
```

Слика 9. Приказ методе за креирање сесије

5. ЗАКЉУЧАК

Рад описује хибридни систем аутентификације и ауторизације (*OAuth 2.0* и *OpenID Connect*) који опслужује апликацију са сесијама, ослања се на *AWS* (*Cognito, DynamoDB, Lambda*) и интегрише се са *.NET/Redis* ресурсном апликацијом. Структура обухвата теоријски увод, постојеће хибридне обрасце, моделе и функционалности система, технологије и њихову конфигурацију, као и примере имплементације и токове аутентификације, ауторизације, промене креденцијала. Кључно ограничење је везаност за једног *cloud* провајдера и зависност од сесијског кода. Даљи рад подразумева енкапсулацију и провајдер-агностичан дизајн ради лакше подршке новим апликацијама.

6. ЛИТЕРАТУРА

[1] Sam Newman. *Building Microservices* (2nd ed.). O'Reilly Media, 2021.
[2] RFC 9700 (BCP 240): *OAuth 2.0 Security Best Current Practice,* 2025.
[5] Fett, D., Küsters, R., Schmitz, G. *A Comprehensive Formal Security Analysis of OAuth 2.0.*
[6] Richer, J., Sanso, A. *OAuth 2 in Action.* Manning Publications, 2017.
[7] Madden, N. *API Security in Action.* Manning Publications, 2020.
[8] Siriwardena, P., Dias, N. *Microservices Security in Action.* Manning Publications, 2020.
[9] RFC 7519 — JSON Web Token (JWT), IETF, 2015.
[10] Michael Wittig, Andreas Wittig. *Amazon Web Services in Action (3rd ed.).* Manning Publications, 2023.
[11] Mark J. Price. *C# 12 and .NET 8 – Modern Cross-Platform Development.* Packt, 2024.

Кратка биографија:

Живко Станишић рођен је 11.08.1995. године у Сомбору. Завршио је основне академске студије 30.10.2019. на Факултету техничких наука у Новом Саду, одсек Рачунарство и аутоматика. Уписао је мастер академске студије. **Контакт**: zivkostanisic@pm.me